UTILITY APPLICATION FOR UNITED STATES PATENT

FOR

APPARATUS AND METHOD FOR CONTROLLING MEMORY ALLOCATION FOR VAIRABLE SIZE PACKETS

Inventor(s):
Bong Wan Kim
Dong Yong Kwak

Blakely, Sokoloff, Taylor & Zafman LLP
12400 Wilshire Boulevard, 7th Floor
Los Angeles, CA 90025
Telephone: (310) 207-3800

APPARATUS AND METHOD FOR CONTROLLING MEMORY ALLOCATION
FOR VARIABLE SIZE PACKETS

This application claims priority from Korean Patent Application No.
2002-81383, filed December 18, 2002, the disclosure of which is incorporated
herein by reference in their entirety.

BACKGROUND OF THE INVENTION

1.   Field of the Invention

The present invention relates to a method of effectively using an electric
memory device (hereinafter referred to as a 'memory'), and more particularly, to
a memory management apparatus and method for storing data in units of
variable size packets in appliances utilizing memory devices.

2.   Description of the Related Art

Generally, when new memory space is needed in a computer or a
communication device, the location of the memory space is determined in part
of the memory device by a process referred to as allocation, and the location is
managed by another memory space which keeps an address value referred to
as a pointer.   For this memory management, most computers and
communications devices employ a method by which a memory device is divided
into fixed-size units and managed.   However, since uniform memory allocation
is performed even for small-sized packets among variable sized packets in this
method, the method has a drawback in that much space in the memory is
wasted.

To solve this problem, a method is suggested in U.S. Pat. No.
6,088,777 granted to Sober on 11 July 2000, entitled "Memory system and
method for dynamically allocating a memory divided into plural classes with
different block sizes to store variable length messages" and other patents,
disclosing a method by which a memory device is not divided into fixed-size
units beforehand but allocated a memory space appropriate to the request
whenever there is a request for allocating a memory space.

However, although this method has an advantage in that wasting memory space can be minimized, its hardware implementation is complicated. In addition, since memory space rearrangement (referred to as garbage collection) should be performed periodically, this method can be applied only by software.

SUMMARY OF THE INVENTION

The present invention provides a memory allocation apparatus and method by which when there is a request for allocating memory space of a variable size, wasting memory space can be minimized by dividing a memory device into a plurality of fixed-size units and managing the memory units so that the management of location address values for memory allocation can be implemented by hardware.

The present invention also provides a computer readable medium having embodied thereon a computer program for executing the memory allocation method in a computer.

According to an aspect of the present invention, there is provided a memory management apparatus comprising: data memory, which comprises a plurality of data blocks, each of which comprises a plurality of sub data blocks having a predetermined size, and when there is a request for allocating memory space of a variable size, allocates memory space in units of any one of the sub data blocks and the data blocks; free list memory, which manages an free memory space of the data memory as at least one or more lists; and registers that store head location information and tail location information of the list.

According to another aspect of the present invention, there is provided a memory allocation method comprising: (a) when n is a power of 2 and i is a positive integer, if the size of a requested memory space for allocation is greater than $\frac{n}{2^i}$ bytes, allocating an $\frac{n}{2^{i-1}}$-byte memory space to a valid entry existing in an $\frac{n}{2^{i-1}}$-byte entry list managed by a free list memory; and (b) if the size of a

2

requested memory space for allocation is equal to or less than $\frac{n}{2^i}$, allocating

an $\frac{n}{2^i}$-byte memory space to a valid entry existing in an $\frac{n}{2^i}$-byte entry list

managed by the free list memory, but if there is no valid entry in the $\frac{n}{2^i}$-byte

entry list, dividing the $\frac{n}{2^{i-1}}$-byte entry list and allocating the divided $\frac{n}{2^{i-1}}$-byte

5 entry list as an $\frac{n}{2^i}$-byte memory space.

According to an aspect of the present invention, there is provided a memory deallocation method comprising: (a) when n is a power of 2 and i is a positive integer, if the size of a deallocated memory space is greater than $\frac{n}{-2^i}$

bytes, deallocating an $\frac{n}{2^{i-1}}$-byte memory space to a data memory and including

10 an entry corresponding to the memory space in an $\frac{n}{2^{i-1}}$-byte entry list managed by a free list memory; and (b) if the size of a deallocated memory space is equal

to or less than $\frac{n}{2^i}$ bytes, deallocating a memory space of $\frac{n}{2^i}$ bytes to the data

memory and including an entry corresponding to the memory space in an $\frac{n}{2^i}$-

byte entry list managed by the free list memory, but if a neighboring memory
15 space managed by the entry which manages the deallocated memory space is not in use, including an entry, which corresponds to a memory space obtained by combining the deallocated memory space and the neighboring memory

space, in the $\frac{n}{2^{i-1}}$-byte entry list.

20   <u>BRIEF DESCRIPTION OF THE DRAWINGS</u>
The above objects and advantages of the present invention will become more apparent by describing in detail exemplary embodiments thereof with

3

reference to the attached drawings in which:

FIG. 1 is a block diagram showing an example of memory space allocation of a memory device according to a preferred embodiment of the present invention in which one data memory block comprises 4 subblocks;

FIG. 2 is a block diagram showing an example of memory space allocation of a memory device according to a preferred embodiment of the present invention in which one data memory block comprises 2 subblocks;

FIG. 3 is a flowchart of the steps performed by a memory space allocation method for the memory device shown in FIG. 2; and

FIG. 4 is a flowchart of the steps performed by a memory space deallocating method for the memory device shown in FIG. 2.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention comprises an apparatus and method for allocating memory space appropriate to memory space allocation requests that are frequently occurring in application programs and communications devices in order to allocate memory space of a variable size in a predetermined region. More specifically, when data is divided into fixed-size units and stored, one data is formed with segments that are each comprised of a header part containing index information and a payload part containing the actually divided fixed-size units. In this case, each segment has a size in a predetermined range. When a request for storing this segment in a memory device is made, the request is made for allocating a memory space less than a maximum size that has been determined beforehand. For managing addresses for memory space allocation, in the prior art, memory is divided into units of a uniform fixed size, or memory allocation is performed by a complicated software-based method. However, in the memory management apparatus and method according to the present invention, memory is divided into units of a plurality of fixed sizes and managed so that the addresses for the allocated memory locations can be managed by hardware. Then, when memory space for storing a variable size packet is allocated, a block or subblock, which is the most suitable for the request for allocation of memory space, is allocated. Accordingly, the

4

apparatus and method are appropriate for high-speed memory space allocation and enable efficient use of the memory space such that wastting memory can be minimized.

FIG. 1 is a block diagram showing an example of memory space allocation of a memory device according to a preferred embodiment of the present invention in which one data memory block comprises 4 subblocks. Referring to FIG. 1, the memory apparatus according to the present invention comprises a free list memory 1100, a data memory 1200, a plurality of registers 1300, and an address translator 1400.

The data memory 1200 is a memory area, in which actual data is to be stored, and comprises a plurality of blocks, each of which comprises a plurality of subblocks having a predetermined length. When a variable size memory space is requested, memory space is allocated in units of the subblocks or in units of the blocks in the data memory 1200. The free list memory 1100 is a memory for managing the data memory 1200 and manages free memory spaces (that is, the memory spaces not in use) with at least one or more lists through the use of pointers. The register 1300 stores the header pointer and tail pointer of a list. The address translator 1400 converts addresses between the data memory 1200 and the free list memory 1100. The memories 1100 and 1200 and the registers 1300 can be implemented in one memory with respective addresses or in their respective memories.

The number of entries of the free list memory 1100 is the same as the number of entries of the data memory 1200. As well, the memories 1100 and 1200 have a 1:1 corresponding relationship. Accordingly, information dealt with in entry number 1 of the free list memory 1100 indicates that this information corresponds to entry number 1 of the data memory 1200. Therefore, a pointer for the corresponding relationship between these two memories is not needed and memory space can be saved.

The address corresponding relationship between the free list memory 1100 and the data memory 1200 will now be explained. For example, if the start address of the data memory 1200 is 0, the start address of each entry forming the data memory 1200 is calculated by a simple rule such as "entry

5

number x block size (n)".   The size of a block of the data memory 1200 can be any number but is usually a number obtained by raising 2 to the power of a positive integer.   Accordingly, the address translator 1400 performs address calculation between the data memory 1200 and the free list memory 1100, by only adding some 0's to an entry number expressed in a binary digit. Particularly, when the process of calculating address between these two memories is implemented by hardware, only simply connecting a signal line corresponding to each entry number to an upper bit line is needed but an address translator is not needed for the implementation.   If the start address of the data memory 1200 is not 0, only adding the start address of the data memory 1200 is needed in the address translator 1400.   That is, in this case, the start address of each entry forming the data memory 1200 is "entry number x block size (n) + start address of data memory."

The free list memory 1100 comprises entries 1101, 1102, ..., with bit masks and pointer values.   Each bit mask corresponds to a subblock included in a block of the data memory 1200 and indicates whether or not the corresponding subblock is now in use.

FIG. 1 shows an example in which when each data memory block 1201, 1202, ..., has four subblocks, each entry 1101, 1102, ..., forming the free list memory 1100 has 4-bit bit masks.   The number of subblocks forming a block 1201, 1202, ..., of the data memory 1200 can be any natural number, but in order to effectively use the present invention to store a variable size packet, a number obtained by raising 2 to the power of a positive integer such as 2, 4, and 8 is useful.

In each entry 1101, 1102, ..., forming the free list memory 1100, a pointer (NEXT_FREE_LIST_PTR) value indicating a next free memory space is basically used.   By using this pointer, the free list memory 1100 maintains the free memory space of the data memory 1200 as a list, and if there is a request for allocation, begins allocation from the header of the list.

The registers 1300 keep a pointer (Header Pointer of Free List; HPFL) for the start location and a pointer (Tail Pointer of Free List; TPFL) for the last location of each list in order to respond to the request for allocation.   For

example, assuming that the number of subblocks in a block is X, the number of pointer pairs of the registers 1300 is "1+$\log_2$X". That is, if the number of subblocks in a block is 4 as in FIG. 1, the registers 1300 have 3 pointer pairs (1 + $\log_2$4 = 3). Accordingly, the registers 1300 have 6 pointers represented as reference numbers 1301 through 1306. At this time, the first and second pointers 1301 and 1302 are used for allocation of memory space of an n-byte size in the data memory 1200. The third and fourth pointers 1303 and 1304 are used for allocation of memory space of an n/2-byte size and the fifth and sixth pointers 1305 and 1306 are used for allocation of memory space of an n/4-byte size in the data memory 1200.

For example, when the size of a block is 256 bytes (n=256) and if there is a request for allocating a memory space of 129 bytes to 256 bytes, an entry corresponding to the value of the first pointer (1HPFL) 1301 is allocated and the value of the first pointer 1301 is updated with a NEXT_FREE_LIST_PTR value. When the allocated 256-byte block is deallocated, the block is added to the free list memory 1100 through the second pointer (1TPFL) 1302. In the same manner, allocating memory space of 65 bytes to 128 bytes is performed through the third pointer (2HPFL) 1303, and deallocating the allocated 128-byte block is performed through the fourth pointer (2TPFL) 1304. Allocating memory space equal to or less than 64 bytes is performed through the fifth pointer (3HPFL) 1305 and deallocating the allocated 64-type block is performed through the sixth pointer (3TPFL) 1306.

At this time, whenever a corresponding data memory block is used, the bit mask indicates whether or not each subblock is allocated and currently used. Accordingly, when an allocated memory space is deallocated, even though 64 bytes are deallocated, if a contiguous subblock in the identical entry is not in use (that is, if a contiguous 128-byte memory space is not used), the subblock is not deallocated as a 64-byte space but is deallocated together with the neighboring 64-byte space as a 128-byte space such that the memory space of the data memory 1200 can be effectively used. For the deallocation operation at this time, instead of the sixth pointer 1306 managing the deallocation of 64 bytes, the fourth pointer 1304 managing deallocation of 128 bytes is used.

7

This deallocation rule of memory space according to the present invention can be applied in the same manner to memory space deallocation between a 128-byte space and a 256-byte space, as well as a 64-byte space and a 128-byte space.

5   As described above, according to the memory management apparatus, the data memory 1200 allocates a block or a subblock of the most suitable size when there is a request for allocation of memory space, and when an allocated memory space is deallocated, deallocates memory space of a maximum size that can be allocated, depending on whether or not a contiguous subblock in the

10   same entry is in use.   Accordingly, the apparatus can effectively respond to a request for allocating a small-sized memory space and optimum utilization of memory space is enabled.

         FIG. 2 is a block diagram showing an example of memory space allocation of a memory device according to an embodiment of the present

15   invention in which one data memory block comprises 2 subblocks.

         Referring to FIG. 2, a data memory 2200 comprises blocks 2201, 2202, ..., each of which has 2 subblocks.   Each entry 2101, 2102, ..., forming a free list memory 2100 corresponding to the data memory 2200, has   2-bit bit masks. In this case, since the number of subblocks of a block in the data memory 2200

20   is 2, the registers have 2 pointer pairs.   Here, the first pointer (1HPFL) 2301 and the second pointer (1TPFL) 2302 stored in the registers 2300 are used for allocation and deallocation of memory space of 128 bytes and the third pointer (2HPFL) 2303 and the fourth pointer (2TPFL) 2304 are used for allocation and deallocation of memory space of 64 bytes.

25   FIG. 2 shows an example of the structure of the free list memory 2100 when the data memory 2200 is actually used.   Referring to FIG. 2, when the bit mask value of the free list memory 2100 is "00", memory space corresponding to bit mask value "00" (that is, a memory space corresponding to entry number 0x0000, 0x0001, or 0x0003) can be allocated as a 128-byte memory space and

30   therefore the free list memory 2100 can form a linked list for a 128-byte memory space as shown in FIG. 2.   In this case, the first pointer 2301 has a value 0 corresponding to the head of the list (that is, entry number 0x0000).

When one of the 2-bit bit masks is 1 and the other bit is 0 (that is, when the bit masks are "10" or "01"), memory space corresponding to the entry (that is, a memory space corresponding to entry number 0x0002, 0x1FFE, or 0x1FFF) can be allocated as a 64-byte memory space and therefore the free list memory 2100 can form a linked list for a 64-byte memory space as shown in FIG. 2. In this case, the third pointer 2303 has a value 2 corresponding to the head of the list (that is, entry number 0x0002), and the fourth pointer 2304 has a value 8091 corresponding to the last value of the list (that is, entry number 0x1FFF). At this time, the previous pointer value (PREV_PTR) of the free list memory 2100 is not necessarily needed, but when there is a previous pointer value (PREV_PTR), there is an advantage in that when the entry is deleted in the list, the entry can be deleted and the list can be reconstructed at the speed of O(1). Here, O(n) is a method to express an execution speed in computer algorithms and indicates an operation speed in proportion to the number n. In addition, $O(n^2)$ indicates a speed in proportion to n squared and O(1) indicates that a time having a predetermined constant value is taken irrespective of the number of entries. For example, when there is no previous pointer value (PREV_PTR) and an entry is to be deleted from a list, O(n) is taken because the list should be reconstructed after searching the list from the start for the entry location (here, n is the number of entries in the list linked by pointers). Compared to this case, when there is a previous pointer value (PREV_PTR) and an entry is to be deleted, the effort to search the list for the entry location is not needed such that the list can be directly reconstructed and O(1) is enabled.

FIG. 3 is a flowchart of the steps performed by a memory space allocation method for the memory device shown in FIG. 2.

Referring to FIG. 3, in order to allocate memory space, it is first determined whether or not the size of a requested memory space to be allocated is equal to or less than 64 bytes in step 3001. If the result of the determination in step 3001 indicates that the size of the requested memory space is equal to or less than 64 bytes, it is determined whether or not the entry of the data memory 2200 corresponding to the third pointer (2HPFL) 2303 of the registers 2300 is valid (that is, whether or not there is an free memory space in

a 64-byte list managed by the free list memory 2100) in step 3002.

If the result of the determination in step 3002 indicates that the entry of the data memory 2200 corresponding to the third pointer (2HPFL) of the registers 2300 is valid, the 64-byte entry corresponding to the third pointer (2HPFL) is allocated as memory space and the bit mask of the corresponding subblock forming the entry is set to 1 in step 3203. Then, the third pointer (2HPFL) is updated with a NEXT_PTR value of the corresponding entry such that the 64-byte memory is allocated in step 3204.

If the result of the determination in step 3002 indicates that the entry of the data memory 2200 corresponding to the third pointer (2HPFL) is not valid, that is, if there is no free memory space in the 64-byte list managed by the free list memory 2100, the 128-byte entry corresponding to the first pointer (1HPFL) is allocated as memory space and the bit mask of the corresponding subblock forming the entry is set to 1 in step 3003. Then, by adding the allocated entry to the fourth pointer (2TPFL) used for managing a 64-byte list, the entry is added to the 64-byte list in step 3004. Next, by updating the first pointer (1HPFL) with the NEXT_PTR value of the corresponding entry, the entry is deleted from the 128-byte list in step 3005. As a result, half of the 128 bytes are allocated as memory space and the remaining half can be used for next 64-byte allocation.

If the result of the determination in step 3001 indicates that the size of the requested memory space is greater than 64 bytes, the 128-byte entry corresponding to the first pointer (that is, 1HPFL) 2301 of the registers 2300 is allocated and the bit mask corresponding to the entry is set to 1 in step 3102. Then, by updating the NEXT_PTR value of the corresponding entry with the first pointer (1HPFL), a 128-byte memory space is allocated in step 3103.

As described above, in the memory space allocation method according to the present invention, a predetermined memory space having a different size is allocated depending on the size of a requested memory space. For example, if the size of the requested memory space to be allocated is greater than 64 bytes, a 128-byte memory space is allocated, and if the size is equal to or less than 64 bytes, a 64-byte memory space is allocated. At this time, the block

size of memory space to be allocated may be further divided by a user. Particularly in the memory space allocation method according to the present invention, when memory space equal to or less than 64 bytes is allocated, and if there is no valid entry in the 64-byte list, an entry in the 128-byte list is divided

5   and then allocated as a 64-byte memory space. Accordingly, the memory allocation method according to the present invention can perform memory space allocation for both 64-byte and 128-byte memory spaces, basically using only the same amount of information as needed in managing 128-byte memory spaces.

10   FIG. 4 is a flowchart of the steps performed by a memory space deallocating method for the memory device shown in FIG. 2, and shows a process for deallocating memory spaces when a block of the data memory 2200 has 2 subblocks.

Referring to FIG. 4, for deallocation of memory spaces, it is first

15   determined whether or not a deallocated memory space is equal to or less than 64 bytes in step 4001. If the result of the determination in step 4001 indicates that the size of the deallocated memory space is equal to or less than 64 bytes, it is determined whether or not the bit masks of the entry of the deallocated memory space are all set to 1 (that is, the corresponding entry can be

20   deallocated so that it can be used as a 128-byte memory space) in step 4002.

If the result of the determination in step 4002 indicates that the bit masks of the deallocated entry are all set to 1 (that is, the corresponding entry is deallocated as a 64-byte memory space), the bit masks corresponding to the deallocated entry are set to 0 and a NEXT_PTR value is set to -1 in step 4203.

25   Then, by setting the deallocated entry as the NEXT_PTR value of the entry corresponding to the fourth pointer (2TPFL) 2304, the entry is added to a 64-byte list in step 4204.

If the result of the determination in step 4002 indicates that the bit masks of the deallocated entry are not all set to 1 (that is, the corresponding

30   entry can be deallocated as a 128-byte memory space), the deallocated entry is deleted from a 64-byte list in step 4003. Then, the bit masks corresponding to the deallocated entry are set to 0 and the NEXT_PTR value is set to -1 in step

11

4004. Next, by setting the deallocated entry as the NEXT_PTR value of the entry corresponding to the second pointer (1TPFL) 2302, the entry is added to a 128-byte list in step 4005.

If the result of the determination in step 4001 indicates that the size of a deallocated memory space is greater than 64 bytes, the bit masks corresponding to the deallocated entry are set to 0, and the NEXT_PTR value is set to -1 in step 4102. Then, by setting the deallocated entry as the NEXT_PTR value of the entry corresponding to the second pointer (1TPFL) 2302, the entry is added to a 128-byte list in step 4103.

As described above, in the memory space deallocation method according to the present invention, predetermined memory spaces having different sizes are deallocated depending on the size of a deallocated memory space. For example, if the size of a requested memory space to be deallocated is greater than 64 bytes, memory space of 128 bytes is deallocated and if the size of a request memory space is equal to or less than 64 bytes, memory space of 64 bytes is deallocated. In the present invention, particularly when memory space equal to or less than 64 bytes is deallocated, if memory space of 64 bytes neighboring the deallocated 64 bytes in the same entry is not in use (that is, if the memory space is free), one 128-byte memory space is deallocated and then used, instead of deallocating the deallocated memory space and the neighboring memory space, as two 64-bytes memory spaces. Accordingly, in the memory space allocation method according to the present invention, a memory space is deallocated always as a maximum size that can be allocated, such that optimum utilization of memory space is enabled.

This memory space allocation and deallocation method provides a structure appropriate to storing Internet Protocol (IP) packets accounting for most of the data transmission on the Internet within the current network equipment. Theoretically, the size of an IP packet (i.e., the size of the header of an IP packet) can be 20 bytes to 64Kbytes. However, average 300-byte packets are mainly used and 40-byte packets are mainly distributed in actual situations.

Accordingly, in most network equipment at present, a packet is divided

12

into fixed segment units of 128 bytes or 256 bytes and then stored. Therefore, the last segment of a packet is mostly wasted, and when the characteristic of an IP packet that mainly uses 40 bytes is considered, if a packet is divided into segment units of 256 bytes or 128 bytes, more than half of the memory space is wasted.

Therefore, if the present invention is applied to network equipment, as shown in FIG. 2, memory space allocation of a 64-byte unit is enabled only with segment management information for 128 bytes, such that the wasting of memory space occurring in a 40-byte packet can be greatly reduced. Particularly, since the present invention can be implemented by hardware as a logic circuit for network equipment that should store packets at a high speed, it is suitable for high-speed memory space allocation and at the same time can utilize memory space effectively.

Though memory space allocation and deallocation for an IP packet in network equipment is specifically shown above as embodiments of the present invention, the present invention can also be applied to allocation and deallocation of memory spaces having different sizes that can be performed in a variety of devices utilizing storage devices (that is, memory devices).

The present invention may be embodied in a code, which can be read by a computer, on a computer readable recording medium. The computer readable recording medium includes all kinds of recording apparatuses on which computer readable data are stored. The computer readable recording media includes storage media such as magnetic storage media (e.g., ROM's, floppy disks, hard disks, etc.), optically readable media (e.g., CD-ROMs, DVDs, etc.) and carrier waves (e.g., transmissions over the Internet). Also, the computer readable recording media can be discributed on computer systems connected through a network and can store and execute a computer readable code in a distributed mode.

According to the memory management apparatus and method for storing various sized packets, memory is divided into units of a plurality of fixed sizes and managed so that the addresses for allocated memory locations can be managed by hardware. Therefore, when memory space for a variable size

13

packet less than a predetermined size is allocated, a block or subblock, which is the most suitable for the request for allocation of memory space, is allocated, and when an allocated memory space is deallocated, memory space of a maximum size that can be allocated is deallocated depending on whether or not a contiguous subblock in the same entry is in use. Accordingly, a request for allocating a small-sized memory space can be effectively managed and optimum utilization of memory space is enabled such that wasting memory can be minimized.